27fw
AF ¢

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Applicant: EDWARDS et al.

Application: METHOD, COMPILER AND PLATFORM INDEPENDENT FRAMEWORK FOR PARSING AND GENERATING DATA STRUCTURES

Serial No.: 09/829,834

Filing Date: April 10, 2001

Art Unit: 2124

Examiner: Trenton J. Roche

Case: ROC920010078US1

CERTIFICATE OF MAILING: I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed for Commissioner of Patents, P.O. Box 1450, Alexandria, VA, 22313-1450, on

June 12, 2006   *Joan Pennington*   Joan Pennington

Date of Deposit   Signature   Name of person signing

---

535 North Michigan Avenue
Unit 1804
Chicago, Illinois 60611

Mail Stop **Appeal Brief-Patents**
Honorable Commissioner Of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF TRANSMITTAL

Sir:

An Appeal Brief for Applicants is being submitted herewith. Please charge the Deposit Account No. 09-0465 of International Business Machine Corporation in the amount of **$500.00** for the fee for filing a brief in support of the appeal (37 CFR §41.20(b)(2) fee code 1402).

-1-

Serial No.: 09/829,834

The Commissioner of Patents and Trademarks is hereby authorized to charge any additional fees or credit any overpayment in connection with the filing of the above-referred to Appeal Brief to the Deposit Account No. 09-0465 of International Business Machine Corporation. A duplicate copy of this transmittal is enclosed.

Respectfully submitted,
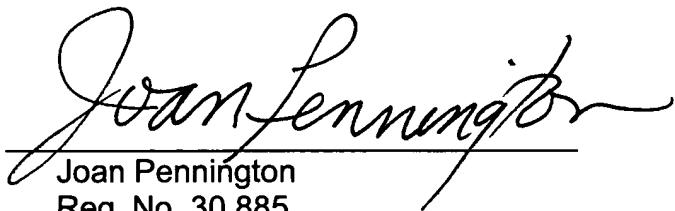
By _____
Joan Pennington
Reg. No. 30,885
Telephone: 312/670-0736
One of the Attorneys for Applicants

Enclosures

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Applicant: EDWARDS et al.

Application: METHOD, COMPILER AND PLATFORM INDEPENDENT FRAMEWORK
FOR PARSING AND GENERATING DATA STRUCTURES

Serial No.:   09/829,834

Filing Date: April 10, 2001

Art Unit: 2124

Examiner: Trenton J. Roche

Case: ROC920010078US1

CERTIFICATE OF MAILING: I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed for Commissioner of Patents, P.O. Box 1450, Alexandria, VA, 22313-1450, on

June 12, 2006 _Joan Pennington_ Joan Pennington

Date of Deposit   Signature   Name of person signing

APPEAL BRIEF FOR APPLICANTS

JOAN PENNINGTON
535 North Michigan Avenue
Unit 1804
Chicago, Illinois 60611

One of the Attorneys for Applicants

TABLE OF CONTENTS

TABLE OF CITATIONS

TABLE OF OTHER AUTHORITIES

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Applicant: EDWARDS et al.

Application: METHOD, COMPILER AND PLATFORM INDEPENDENT FRAMEWORK
FOR PARSING AND GENERATING DATA STRUCTURES

Serial No.:   09/829,834

Filing Date: April 10, 2001

Art Unit: 2124

Examiner: Trenton J. Roche

Case: ROC920010078US1

CERTIFICATE OF MAILING: I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed for Commissioner of Patents, P.O. Box 1450, Alexandria, VA, 22313-1450, on

June 12 2006   Joan Pennington

Date of Deposit   Signature   Name of person signing

---

535 North Michigan Avenue
Unit 1804
Chicago, Illinois 60611

Mail Stop **Appeal Brief Patents**
Honorable Commissioner Of Patents
P.O Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF FOR APPLICANTS

Sir:

This is an appeal of the final rejection of claims 1 and 10-14 under 35

U.S.C. §102(b) and claims 2-9 under §103(a) mailed November 1, 2005. For the

reasons set forth below, it is submitted that the Board should reverse the final rejection

of claims 1-14.

(1) <u>REAL PARTY IN INTEREST</u>

The real party of interest is International Business Machines Corporation.

(2) <u>RELATED APPEALS AND INTERFERENCES</u>

Applicants' attorney knows of no other appeals or interferences that would have a bearing on the Board's decision in the present appeal.

(3) <u>STATUS OF CLAIMS</u>

Claims 1and 10-14 have been finally rejected under 35 U.S.C. § 102(b) and claims 2-9 have been finally rejected under as unpatentable under 35 U.S.C. § 103(a) in an office action mailed November 1, 2005. The rejections of each of the pending claims 1-14 have been appealed.

(4) <u>STATUS OF AMENDMENTS</u>

No amendment was filed after the final rejection of claims.

(5) <u>SUMMARY OF CLAIMED SUBJECT MATTER</u>

The claimed invention as recited by independent claims 1, 10, and 14, and representative dependant claims 2, 5, and 7 can best be appreciated and understood with reference to the patent specification (hereinafter page p., line l.) and drawings of the invention, attached in (9) Evidence Appendix (Sheets 1-13).

The present invention effectively implements a method, compiler and platform independent framework for parsing and generating data structures.

As presented independent claim 1 recites a computer-implemented method for parsing and generating data structures for use by data processing applications in a computer system comprising the steps of: utilizing sizeof and offsetof functions, defining

a length and a location of each parameter of a data structure; and storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures.. (p. 2, l. 17-21).

As presented independent claim 10 recites a compiler and platform independent framework for parsing and generating data structures used by data processing applications in a computer system comprising: means for defining a length and a location of each parameter of a data structure utilizing sizeof and offsetof functions; and means for storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures. (p. 2, l. 17-21).

As presented independent claim 14 recites a computer program product for parsing and generating data structures for use by data processing applications in a computer system, said computer system having a processor; a memory controller coupled to said processor by a system bus; a main memory coupled to said memory controller; said computer program product including a plurality of computer executable instructions stored on a computer readable medium, wherein said instructions, when executed by said computer system, cause said computer system to perform the steps of: utilizing sizeof and offsetof functions, defining a length and a location of each parameter of a data structure; and storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures. (p. 2, l. 17-21).

In accordance with features of the invention, the identifier structure is based on the definition of the data structure itself so that the problem of duplicating the data structure definition is eliminated. The sizeof and offsetof functions automatically account for compiler and platform differences which otherwise would lead to alignment problems so that compiler and platform independence from data structure realignment is provided. (p. 2, l. 22-27).

Having reference now to the drawings, in FIG. 1, there is shown a computer or data processing system generally designated by the reference character 100 for carrying out the methods for parsing and generating data structures of the preferred embodiment. As shown in FIG. 1, computer system 100 includes a central processor unit (CPU) 102 connected by a system bus to a memory controller 103, a main memory 104 including an operating system 104A and applications 104B, and a display adapter 106 coupled to a display 108. CPU 102 is connected to a user interface (UI) adapter 110 connected to a pointer device and keyboard 112. CPU 102 is connected to an input/output (IO) adapter 114 connected to a direct access storage device (DASD) 116 and a tape unit 118. CPU 102 is connected via an Asynchronous Transfer Mode (ATM) card communications adapter 120 to an ATM network 122. It should be understood that the present invention is not limited to a computer model with a single CPU, or other single component architectures as shown in FIG. 1. (p. 5, l. 2-16).

Central processor unit 102 is suitably programmed for generating data structures and information to manipulate the data, such as illustrated and described with respect to FIGS. 3-8 and 10-13 and to execute the flowchart of FIGS. 14A and 14B. Computer

100 may be implemented using any suitable computer, such as an IBM personal computer running the OS/2® operating system. (p. 5, l. 17-22).

In accordance with features of the preferred embodiment, a method is provided that can be used by procedural table-driven or object oriented rules-driven approaches for generating and parsing data structures. A pair of functions, sizeof() and offsetof() functions are used. The sizeof() and offsetof() functions are built into the C and C++ programming languages. It should be understood that other languages have sizeof() and offsetof() functions or these functions can be built within the constraints of other languages. In accordance with features of the preferred embodiment, instead of implementing a table or rule object as a redundant definition of the data structure, the length and location of each of the data structure's parameters are defined within the table or rule object by the sizeof() and offsetof() functions. Thus, the table or rule object is based on the definition of the data structure itself. This differs from treating the data structure purely as a string of bytes in that the offsetof() function provides the location of each parameter within the structure whereas a purely string implementation would only deal with the length of each parameter. Additionally, the sizeof() and offsetof() functions automatically account for compiler and platform differences which otherwise would lead to alignment problems. Since the sizeof() and offsetof() functions execute at compile time, no performance penalty results. (p. 5, l. 23 - p.6, l.8).

FIG. 2A illustrates in more detail software components of a user application 104B to utilize the ATM card communications adapter 120 in the computer system 100 in accordance with the preferred embodiment. User application 104B is used with a

LAN emulation layer, an SVC (Q.2931), a data link layer (Q.SAAL), and a physical layer. User application 104B of the preferred embodiment is the call control layer for the ATM SVC (Q.2931). this protocol consists of a number of messages, SETUP, CALL PROCEEDING, CONNECT, RELEASE, and the like, to establish and clear switched virtual calls. Each of these messages is comprised of a number of Information Elements (IEs). There are dozens of different IEs defined by the Q.2931 standard and each ID contains from a few to many, for example, up to 20 separate data parameters. The user interface to the call control layer contains a similar degree of complexity and the same number of data parameters. The magnitude of individual data parameters which need to be handled at both the user and network interfaces make this ATM call control layer a prime candidate for a table-based or rule object-based implementation. (p. 6, l. 9-25).

FIGS. 2B and 2C respectively illustrate a Connection Identifier Information Element (IE) generally designated by the reference character 200 and AtmSvcConnIdleRules generally designated by the reference character 202 for parsing and generating data structures in accordance with the preferred embodiment. An object oriented rules approach is applied to parsing and generating the network-side IE and then the object oriented rule is extended to additionally parse and generate the user-side data structure. AtmSvcConnIdleRules 202 illustrate the method for parsing and generating data structures of the preferred embodiment is applied to one IE. The Connection Identifier IE 200 begins with a 4-byte header whose format is generic to all IEs. The next byte contains a 2-bit constant, VP Associated Signaling and a 3-bit

parameter, Preferred/Exclusive. The next two bytes comprise the Virtual Path

Connection Identifier (VPCI) parameter. The final two bytes comprise the Virtual

Channel Identifier (VCI) parameter. The network layer software also passes these

parameters across the user-side API. FIG. 2B illustrates the format and content of the

Connection Identifier IE 200. (p. 6, l. 26- p. 7, l. 7).

The following rule object for the Connection Identifier IE 200 and its

corresponding constructor contain all of the information unique to this IE and thus is all

of the code unique to parsing and formatting it. (p. 7, l. 8-10).

```
Class AtmSvcConnIdleRule: public AtmSvcIeRule
{
public:
        AtmSvcConnIdleRules():   // Constructor for this class
protected:                       // protected member data
        // The following three data members represent the data parameters
        // passed between the network and the user of the ATM SVC layer.
        //
        // The first of these is of the SubByteData class.  SubByteData
        // is a class which contains methods (parse, format, read from
        // memory, write to memory) for dealing with data in IEs which is less       // than
one byte long.  There may be more than one of these data
        // entities contained within a single byte. (See grp 5: an Extension bit
        // Group 204 in FIG. 2C.)
        //
        // The last two data parameters are of the TwoByteData class.
        // TwoByteData is a class which contains methods (parse, format,
        // read from memory, write to memory) for dealing with data in IEs
        // which is exactly two bytes long.  (See grp 6: a fixed length group
        // 206 and grp 7 a fixed length group 208 in FIG. 2C.)
```

```
        SubByteData prefExclConnIdlePram_;
        TwoByteData vpciConnIdlePram_;
        TwoByteData vciConnIdlePram_;
private:
        enum {ield = CONNID_IE_ID   } // 0X5A = value of the IE identifier
}; (p. 7, l.11 - p. 8, l. 2).


        AtmSvcConnIdleRules: : AtmSvcConnIdleRules()
AtmSvcIeRule(ield)
{
        // The byte in this IE immediately following the 5 bytes of header, is
        // in the format known as an extension bit group.  In this case,
        // the data is a mandatory field in the IE.
        static          ExtBitGroup grp5 //(204 in FIG. 2C, this
                                          MANDATORY_FLD,
                                          0x08);  // initialize the byte to OX08;
        // the 3-bit preferred/exclusive parameter is contained in this
        // extension bit group.
        prefExclConnIdleParm_.Init (this
                                 0x07);  // mask defining location


        // Define the 3 allowed values for preferred/exclusive
        static CheckSpecificValues       prefExclValidityCheck (this);
        prefExclValidityCheck.DefineCodePoint (0x00);
        prefExclValidityCheck.DefineCodePoint (0x01);
        prefExclValidityCheck.DefineCodePoint (0x04);


        // The next defined bytes in the IE is the two byte VPCI parameter.
        // This is defined to be a fixed length group
        static   FixedLenGroup grp6 //(206 in FIG. 2C, this
                                          MANDATORY_FLD,
        vpciConnIdleParm_.Init (this);
```

```
// Define the range of valid values for the VPCI
static CheckRangeOfVals          vpciValidityCheck (this);
vpciValidityCheck.DefineMinValue (0);
vpciValidityCheck.DefineMaxValue (255);

// The last two bytes in the Connection Id IE is the two byte VCI
// parameter.
static  FixedLenGroup grp7 //(208 in FIG. 2C, this
                                          MANDATORY_FLD,
vciConnIdleParm_.Init (this);

// Define the range of valid values for the VCI
static CheckRangeOfVals          vciValidityCheck (this);
vpciValidityCheck.DefineMinValue (32);
vpciValidityCheck.DefineMaxValue (65535);  (p. 8, l. 3-p. 9, l. 7).
```

The following structure is used by the user-side API and contains the three data parameters associated with the Connection Identifier IE 200.   (p. 9, l. 8-9).

```
        struct          conn_id_struct
{
        uint8  prefd_exclus;
        uint16 vpci;
        uint16 vci;
}   (p. 9, l. 10-15).
```

The following change to the constructor for the AtmSvcConnIdleRules object makes use of the method of the preferred embodiment and thereby enhances this rule object so that it describes the Connection Identifier information on both the network side and the user side. (p. 9, l. 16-19).

```
        AtmSvcConnIdleRules: : AtmSvcConnIdleRules()
```

```
:AtmSvcleRule(ield)
{
        conn_id_struct      connidUserStruct; // temporary variable needed
                                        // for the sizeof function;
        // The byte in this IE immediately following the 5 bytes of header, is
        // in the format known as an extension bit group.  In this case,
        // the data is a mandatory field in the IE.
        static      ExtBitGroup grp5 //(204 in FIG. 2C, this
                                        MANDATORY_FLD,
                                        0x08);  // initialize the byte to OX08;
        // the 3-bit preferred/exclusive parameter is contained in this
        // extension bit group.
        prefExclConnIdleParm_.Init (this
                                0x07);  // mask defining location


        // Store the location and length preferred/exclusive parameter
        // relative to the conn_id_struct into the prefExclConnIdleParm_
        // object.
        prefExclConnIdleParm_
        .InitUserStructInfo (   offsetof(conn_id_struct, prefd_exclus ),
                                sizeof(connidUserStruct.prefd_exclus ) );
        // Define the 3 allowed values for preferred/exclusive
        static CheckSpecificValues      prefExclValidityCheck (this);
        prefExclValidityCheck.DefineCodePoint (0x00);
        prefExclValidityCheck.DefineCodePoint (0x01);
        prefExclValidityCheck.DefineCodePoint (0x04);


        // The next defined bytes in the IE is the two byte VPCI parameter.
        // This is defined to be a fixed length group
        static   FixedLenGroup grp6 //(206 in FIG. 2C, this
                                        MANDATORY_FLD,
        vpciConnIdleParm_.Init (this);
```

```
// Store the location and length vpci parameter
// relative to the conn_id_struct into the prefExclConnIdleParm_
// object.
prefExclConnIdleParm_
.InitUserStructInfo (   offsetof(conn_id_struct, vpci  ),
                        sizeof(connidUserStruct.vpci  )  );


// Define the range of valid values for the VPCI
static CheckRangeOfVals          vpciValidityCheck (this);
vpciValidityCheck.DefineMinValue (0);
vpciValidityCheck.DefineMaxValue (255);


// The last two bytes in the Connection Id IE is the two byte VCI
// parameter.
static   FixedLenGroup grp7 //(208 in FIG. 2C, this
                              MANDATORY_FLD,
vciConnIdleParm_.Init (this);


// Store the location and length vci parameter
// relative to the conn_id_struct into the prefExclConnIdleParm_
// object.
prefExclConnIdleParm_
.InitUserStructInfo (   offsetof(conn_id_struct, vci  ),
                        sizeof(connidUserStruct.vci  )  );


// Define the range of valid values for the VCI
static CheckRangeOfVals          vciValidityCheck (this);
vpciValidityCheck.DefineMinValue (32);
vpciValidityCheck.DefineMaxValue (65535);  (p. 9, l. 20- p. 11, l. 12).
```

Execution of the above version of the constructor results in an object hierarchy depicted

in FIG. 2C. An example showing a method of build, that is formatting a PDU or

Connection Identifier IE 200 directly from an API data structure or conn_id_struct is illustrated and described with respect to FIGS. 14A and 14B. (p. 11, l. 13 - 17).

FIG. 2D illustrates an exemplary arrangement generally designated by the reference character 210 of a compiler and platform independent framework 212 for parsing and generating data structures in accordance with the preferred embodiment with exemplary source code 214 and CPU memory 104. As shown in FIG. 2, exemplary API source code 214 includes the following: (p. 11, l. 18 - 23).

```
        struct conn_id_struct
{
        char            prefd_exclus;
        short int       vpci;
        short int       vci;
} (p. 11, l. 24 - 29).
```

FIG. 3 illustrates a memory view of a compiler generating packed data structures generally designated by the reference character 300 in accordance with the preferred embodiment. (p. 11, l. 30 - 32)

FIG. 4 illustrates information generally designated by the reference character 400 to manipulate the data generated by the compiler for the packed data structures 300 in accordance with the preferred embodiment. As shown, structure field prefd_exclus has a sizeof() equal to 1 byte and an offsetof () equal to 0. Structure field vpci has a sizeof() equal to 2 bytes and an offsetof () equal to 1. Structure field vci has a sizeof() equal to 2 bytes and an offsetof () equal to 3. (p. 12, l. 1-7)

FIG. 5 illustrates a memory view of a compiler generating data structures using

16-bit alignment generally designated by the reference character 500 in accordance with the preferred embodiment. Data structure 500 includes wasted or not used memory indicated by x. Data structure 500 avoids split of vci over a split boundary. (p. 12, l. 8-12)

FIG. 6 illustrates information generally designated by the reference character 600 to manipulate the data generated by the compiler for the packed data structures 500 in accordance with the preferred embodiment. As shown, structure field prefd_exclus has a sizeof() equal to 1 byte and an offsetof () equal to 0. Structure field vpci has a sizeof() equal to 2 bytes and an offsetof () equal to 2. Structure field vci has a sizeof() equal to 2 bytes and an offsetof () equal to 4. (p. 12, l. 13-19)

FIG. 7 illustrates a memory view of a compiler generating data structures using 32-bit alignment generally designated by the reference character 700 in accordance with the preferred embodiment. Data structure 700 is used where 4-byte memory read is most efficient and is set to 4-byte boundary. Data structure 700 includes wasted or not used memory indicated by x. (p. 12, l. 20-25)

FIG. 8 illustrates information generally designated by the reference character 800 to manipulate the data generated by the compiler for the packed data structures 700 in accordance with the preferred embodiment. As shown, structure field prefd_exclus has a sizeof() equal to 1 byte and an offsetof () equal to 0. Structure field vpci has a sizeof() equal to 2 bytes and an offsetof () equal to 4. Structure field vci has a sizeof() equal to 2 bytes and an offsetof () equal to 8. (p. 12, l. 26-32)

FIG. 9 illustrates another exemplary source code structure generally designated

by the reference character 900 in accordance with the preferred embodiment. An

advantage of the preferred embodiment is provided when the same source code is to

be compiled for different platforms. Exemplary source code structure 900 is modified to

contain a pointer field. As shown in FIG. 9, exemplary source code 900 includes the

following: (p. 13, l. 1-6)

```
        struct conn_id_struct
{
        void*        next;
        char         prefd_exclus;
        short int    vpci;
        short int    vci;
} (p. 13, l. 7-13)
```

FIG. 10 illustrates a memory view of a compiler generating aligned data

structures for 32-bit CPU architecture generally designated by the reference character

1000 for the exemplary source code 900 in accordance with the preferred embodiment.

Wasted memory space in data structure 1000 is indicated by x. (p. 13, l. 14-18)

FIG. 11 illustrates information generally designated by the reference character

1100 to manipulate the data generated by the compiler for the aligned data structures

1000 in accordance with the preferred embodiment. As shown, structure field next has

a sizeof() equal to 4 bytes and an offsetof () equal to 0. Structure field prefd_exclus has

a sizeof() equal to 1 byte and an offsetof () equal to 4. Structure field vpci has a sizeof()

equal to 2 bytes and an offsetof () equal to 6. Structure field vci has a sizeof() equal to

2 bytes and an offsetof () equal to 8. (p. 13, l. 19-26)

FIG. 12 illustrates a memory view of a compiler generating aligned data

structures for 64-bit CPU architecture generally designated by the reference character

1200 in accordance with the preferred embodiment. (p. 13, l. 27-29)

FIG. 13 illustrates information generally designated by the reference character

1300 to manipulate the data generated by the compiler for the aligned data structures

1200 in accordance with the preferred embodiment. As shown, structure field next has

a sizeof() equal to 8 bytes and an offsetof () equal to 0. Structure field prefd_exclus has

a sizeof() equal to 1 byte and an offsetof () equal to 8. Structure field vpci has a sizeof()

equal to 2 bytes and an offsetof () equal to 10. Structure field vci has a sizeof() equal to

2 bytes and an offsetof () equal to 12. (p. 13, l. 30- p. 14, l. 1-5)

Referring now to FIGS. 14A and 14B, there are shown flow charts illustrating

sequential steps for build of a PDU, such as connection identifier IE 200, directly from

an API data structure in accordance with the preferred embodiment. As shown in this

case, the API data structure is the conn_id_struct. Referring also to FIG. 2C, a header

is built using the IE identifier as indicated in a block 1400. Next a first group rule is

obtained as indicated in a block 1402. A first data rule in the group is obtained as

indicated in a block 1404. Checking for a type of group is performed as indicated in a

block 1406. When an extension bit group is identified, such as grp 5: Extension bit

group 204, the PDU byte is set to an initial value, such as 0x08, as indicated in a block

1408. After the PDU byte is set to an initial value or when a fixed length group is

identified, such as grp 6 fixed length group 206 or grp 7: fixed length group 208, then

the first data rule is obtained as indicated in a block 1410. Checking for a type of data

rule is performed as indicated in a decision block 1412. When two-byte data is

identified for the data rule, then the data is read from the API structure using length and offset stored in this data rule as indicated in a block 1414. Also this data is written into the next two-bytes of the PDU and the PDU buffer pointers are advanced at block 1414. When sub-byte data is identified for the data rule, then the data is read from the API structure using length and offset stored in this data rule as indicated in a block 1416. This data is written into the bits defined by mask stored in this data rule and the PDU buffer pointers are advanced if this was the last data rule for this group at block 1416. (p. 14, l. 6-29)

Then the sequential operations continue following entry point A in FIG. 14B to get the next data rule as indicated in a decision block 1418. Then the sequential operations continue following entry point B in FIG. 14A to identify the type of the group rule at decision block 1406. Otherwise, if the next data rule is null, then the next group rule is obtained as indicated in a decision block 1406. Then the sequential operations continue following entry point C in FIG. 14A to get the first data rule in the group rule at block 1404. Otherwise, if the next group rule is null, then the sequential operations end as indicated in a block 1422. (p. 14, l. 30- p. 15, l. 3)

It should be understood that the compiler and platform independent framework for parsing and generating data structures 212 of the preferred embodiment is not limited to ATM call control or data communications. For example, the method for parsing and generating data structures of the preferred embodiment advantageously can be used with various applications, such as control of writing and reading data storage in disk, tape, or the like. (p. 15, l. 4-10)

Referring now to FIG. 15, an article of manufacture or a computer program product 1500 of the invention is illustrated. The computer program product 1500 includes a recording medium 1502, such as, a floppy disk, a high capacity read only memory in the form of an optically read compact disk or CD-ROM, a tape, a transmission type media such as a digital or analog communications link, or a similar computer program product. Recording medium 1502 stores program means 1504, 1506, 1508, 1510 on the medium 1502 for carrying out the methods for parsing and generating data structures of the preferred embodiment in the system 100 of FIG. 1. (p. 15, l. 11-19)

A sequence of program instructions or a logical assembly of one or more interrelated modules defined by the recorded program means 1504, 1506, 1508, 1510, direct the computer system 100 for parsing and generating data structures of the preferred embodiment. (p. 15, l. 20-23)

(6) <u>GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL</u>

The first ground of rejection presented for review is the rejection of claims 1 and 10-14 under 35 USC §102(b) as being anticipated by Adusumilli, U.S. patent 5,870,749.

The second ground of rejection presented for review is the rejection of claims 2-9 under 35 USC §103(a) as being unpatentable over Adusumilli, U.S. patent 5,870,749 in view of Sasagawa et al., U.S. patent 6,028,863.

(7) <u>ARGUMENT</u>

A. <u>INTRODUCTION</u>

Referring now to FIG. 15, an article of manufacture or a computer program product 1500 of the invention is illustrated. The computer program product 1500 includes a recording medium 1502, such as, a floppy disk, a high capacity read only memory in the form of an optically read compact disk or CD-ROM, a tape, a transmission type media such as a digital or analog communications link, or a similar computer program product. Recording medium 1502 stores program means 1504, 1506, 1508, 1510 on the medium 1502 for carrying out the methods for parsing and generating data structures of the preferred embodiment in the system 100 of FIG. 1. (p. 15, l. 11-19)

A sequence of program instructions or a logical assembly of one or more interrelated modules defined by the recorded program means 1504, 1506, 1508, 1510, direct the computer system 100 for parsing and generating data structures of the preferred embodiment. (p. 15, l. 20-23)

(6) <u>GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL</u>

The first ground of rejection presented for review is the rejection of claims 1 and 10-14 under 35 USC §102(b) as being anticipated by Adusumilli, U.S. patent 5,870,749.

The second ground of rejection presented for review is the rejection of claims 2-9 under 35 USC §103(a) as being unpatentable over Adusumilli, U.S. patent 5,870,749 in view of Sasagawa et al., U.S. patent 6,028,863.

(7) **ARGUMENT**

A. INTRODUCTION

The claims 1-14 on appeal do not all stand or fall together. The claims may conveniently be separately considered based upon the recited subject matter. The independent claims at issue here are claims 1, 10, and 14, and representative dependant claims 2, 5, and 7 are separately patentable.

Applicants respectfully submit that the Examiner's rejections under 35 USC §102(b) and under 35 U.S.C. § 103(a) should be reversed because the subject matter of each of independent claims 1, 10, and 14 and representative claims 2, 5, and 7 is patentable over all the references of record.

Further there is no teaching or suggestion in any of the cited references, individually or taken as a whole, to make the claimed invention obvious.

The rejections of the pending claims under 35 USC §102(b) and under 35 U.S.C. §103(a) are improper and should be reversed.

Applicants respectfully submit that each of the independent claims 1, 10, and 14, and representative dependant claims 2, 5, and 7, is clearly patentable over all the references of record including Adusumilli and Sasagawa et al.

Applicants respectfully submit that considering the subject matter as a whole of the claimed invention as recited in each of the independent claims 1, 10, and 14 requires a conclusion that all the claims 1-14 are patentable.

B. THE SCOPE AND CONTENT OF THE PRIOR ART

Adusumilli, U.S. patent 5,870,749

Adusumilli, U.S. patent 5,870,749 discloses a method for translating attribute data carried in Common Management Information Protocol (CMIP) Protocol Data Units (PDUs) to/from custom designed data structures. A supplementary method for incorporating user's preferences on the data structures and the relationships between different fields in these data structures and the corresponding attribute values is also provided. The translation method automatically performs conversions between the user-designed data structures and various CMIP requests/responses automatically, and, in accordance with user's preferences. The method allows users to simplify and/or compact the storage representation of the Managed Objects by taking advantage of application specific knowledge, and by eliminating unnecessary fields from CHOICE data types in the target data structures. Benefits of the methods presented in this disclosure include automatic translation of CMIP PDUs to/from user-designed data structures, ability to store Managed Object data in space-efficient manner, and automatic generation of data structures for use in communicating with devices using proprietary data representation. At column 11, lines 18-50 states:

"The MOClassTable shown in FIG. 4 contains an entry for each configured Managed Object Class. The localFormClassId field contains the local-form identifier (an integer value) specified using the LOCAL-ID clause of the CLASS configuration. This may be used internally in place of the object-identifier of the class. The CStructureName corresponds to the name of the C structure specified with the DATA-TYPE clause of the CLASS configuration. By

default the configuration program derives this name from the managed object class name (for example by adding a prefix and capitalizing the first letter of the class name). The user can set this to his/her own data structure name, if desired. The CStructureSize field contains the size of the C structure specified in the DATA-TYPE clause. This is only set in the translation tables (the sizeof()operator may be used to compute this size), and is not used in the configuration stage. The ClassInfoPointer field is used to store a link to the meta data generated by the GDMO compiler for this managed object class definition. The ClassAttributeTablePointer points to a class-specific attribute table that contains an entry for each attribute included in the ATTRIBUTES clause of the CLASS specification. The localFormAttributeId field of each ClassAttributeTable entry contains the local-form identifier value of the attribute. This value is copied from the corresponding entry in the GlobalAttributeTable at the time of generating translation tables or a new configuration file. The fieldName field is set to the field-name specified for this attribute in the ATTRIBUTES clause, or to the attribute-label if the field-name is omitted. The fieldOffset field is set to the offset of the corresponding field in the C structure associated with the managed object class. This field is set only in the translation tables (the offsetof() macro may be used to compute the offset of the field) and is not used in the configuration stage."

Sasagawa et al., U.S. patent 6,028,863

Sasagawa et al., U.S. patent 6,028,863 discloses a device at the terminal unit and a device at the network that support an interim local management (ILMI) protocol. When the power is applied to the device at the terminal unit, it notifies the device at the network of support range information about a VPI/VCI of the device at the terminal unit. The device at the network assigns a VPI/VCI to the device at the terminal unit according

to the support range information about the VPI/VCI received in a cold start trap message from the device at the terminal unit when a signal is received from the device at the terminal unit. FIG. 43 shows the data format for use in specifying the connection identifier contained in the signaling message used in the fifth preferred embodiment of the present invention. It shows the details of the element (14) of each message shown in FIGS. 29 through 31. In FIG. 43, the field "virtual path connection identifier" stores the VPCI (corresponding to the VPI), and the field "virtual channel identifier" stores the VCI. The invariable indication field "preferred/exclusive" stores a 3-bit value having one of the following meanings. 000: VPCI is invariable, and VCI is also invariable. 001: VPCI is invariable, but VCI is variable. 010: VPCI is variable, but VCI is invariable.

C. THE REJECTION OF CLAIMS 1 and 10-14 AS BEING ANTICIPATED BY ADUSUMILLI SHOULD BE REVERSED

Anticipation is a question of fact. In re King, 801 F.2d 1324, 231 USPQ 136 (Fed. Cir. 1986). The inquiry as to whether a reference anticipates a claim must focus on what subject matter is encompassed by the claim and what subject matter is described by the reference. As set forth by the court in Kalman v. Kimberly-Clark Corp., 713 F.2d 760, 218 USPQ 781, 789 (Fed. Cir. 1983), cert. denied, 465 U.S. 1026 (1984), it is only necessary for the claims to "'read on' something disclosed in the reference, i.e., all limitations in the claim are found in the reference, or 'fully met' by it." Anticipation under § 102 can be found only when the reference discloses exactly what is claimed; where there are differences between the reference disclosure and the claim, the rejection must be based on § 103 which takes differences into account. Tyler

Refrigeration v. Kysor Industrial Corp., 777 F.2d 687, 689, 227 U.S.P.Q. 845 846-47 (Fed. Cir. 1985). It must be shown that the reference contains all of the elements of the claims, and that the elements are arranged in the same way to achieve the same result which is asserted to be an inventive function.

Claim 1 is patentable

Independent claim 1 recites a computer-implemented method for parsing and generating data structures for use by data processing applications in a computer system comprising the steps of: utilizing sizeof and offsetof functions, defining a length and a location of each parameter of a data structure; and storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures.

Applicants respectfully submit that subject matter of the invention, as recited in independent claim 1, is not anticipated by Adusumilli. Only Applicants teach utilizing sizeof and offsetof functions, defining a length and a location of each parameter of a data structure; and storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures. Adusumilli does not disclose, expressly or under principles of inherency, each and every element of a claimed invention as recited in claim 1. The rejection of claim 1 under 35 U.S.C. § 102(b) is improper and should be reversed.

In accordance with features of the invention, only Applicants teach instead of

implementing a table or rule object as a redundant definition of the data structure, the length and location of each of the data structure's parameters are defined within the table or rule object by the sizeof() and offsetof() functions. Thus, the table or rule object is based on the definition of the data structure itself. This claimed feature of the invention is not disclosed, nor remotely suggested in the Adusumilli patent.

In accordance with features of the invention, only Applicants teach that the identifier structure is based on the definition of the data structure itself, so that the problem of duplicating the data structure definition is eliminated. This feature of the invention is not disclosed, nor remotely suggested in the Adusumilli patent.

This differs from treating the data structure purely as a string of bytes in that the offsetof() function provides the location of each parameter within the structure whereas a purely string implementation would only deal with the length of each parameter. Additionally, the sizeof() and offsetof() functions automatically account for compiler and platform differences which otherwise would lead to alignment problems. Since the sizeof() and offsetof() functions execute at compile time, no performance penalty results.

Applicants acknowledge that the sizeof() and offsetof() functions are built into the C and C++ programming language. Applicants acknowledge that sizeof and offsetof functions are known in the art, such as disclosed by Adusumilli. The Examiner quotes Adusumilli: "The CStructureSize field contains the size of the C structure..." however, the Examiner fails to refer to any teaching or suggestion in Adusumilli for defining a length and a location of each parameter of a data structure. Applicants respectfully

submit that there is no teaching or suggestion in Adusumilli for <u>defining a length and a location of each parameter of a data structure</u>, as taught and claimed by Applicants.

Further the Examiner fails to identify any teaching or suggestion in Adusumilli that provides for storing said length and said location <u>of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures</u>, as taught and claimed by Applicants.

Applicants respectfully submit that Adusumilli does not enable, nor provide any suggestion of parsing and generating data structures by utilizing sizeof and offsetof functions, defining a length and a location of each parameter of a data structure; and storing said length and said location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures, as taught and claimed by Applicants.

Adusumilli teaches a method for generating translation tables includes the steps of parsing the input configuration file and building the MOClassTable and ClassAttributeTable using the CLASS specifications in the configuration file, building the GlobalAttributeTable using the ATTRIBUTE specifications in the configuration file, and building the ASN1TypeTable using the TYPE specifications in the configuration file, propagating the configuration information for not yet configured Managed Object Classes by copying the configuration information from their super classes, propagating the ASN.1 Type configuration to attributes that have complex syntax, but not yet configured, and generation of translation tables (MOClassTable, ClassAttributeTables, GlobalAttributeTable, and ASN1TypeTable) containing this information in a form

convenient for usage by the translator library module.. Contrary to the Examiner's assertion, the use of the sizeof() operator and the offsetof() of Adusumilli is not equivalent to and does not suggest, defining a length and a location of each parameter of a data structure as taught and claimed by Applicants. Contrary to the Examiner's assertion, the use of the sizeof() operator and the offsetof() of Adusumilli is not equivalent to and does not suggest, storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures.

Thus, an independent claim 1 is patentable.

Independent claim 10 is patentable

Each of the independent claim 10 and dependent claims 11, 13, and 13 is submitted to be patentable for the same reasons set forth above in connection with claim 1. Independent claim 10 recites a compiler and platform independent framework for parsing and generating data structures used by data processing applications in a computer system comprising: means for defining a length and a location of each parameter of a data structure utilizing sizeof and offsetof functions; and means for storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures.

The recited means for storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures is not shown, nor suggested,

by Adusumilli relied upon by the Examiner. Further a combination of all the teachings of the references of record would not achieve the claimed invention as recited by claim 10.

Adusumilli does not suggest any means for storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures. For a claim of a patent to be "anticipated" each and every element of that claim must be disclosed in a single prior art reference. Independent claim 10 is not anticipated by Adusumilli.

Thus, independent claim 10 is patentable.

Independent claim 14 is patentable

Independent claim 14 is submitted to be patentable for the same reasons set forth above in connection with claim 1. Independent claim 14 recites a computer program product for parsing and generating data structures for use by data processing applications in a computer system, said computer system having a processor; a memory controller coupled to said processor by a system bus; a main memory coupled to said memory controller; said computer program product including a plurality of computer executable instructions stored on a computer readable medium, wherein said instructions, when executed by said computer system, cause said computer system to perform the steps of: utilizing sizeof and offsetof functions, defining a length and a location of each parameter of a data structure; and storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures.

The recited instructions of the computer program product of the invention, when executed by said computer system, cause said computer system to perform the steps of: utilizing sizeof and offsetof functions, defining a length and a location of each parameter of a data structure; and storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures is not shown, nor suggested, by Adusumilli. Also a combination of all the teachings of the references of record would not achieve the claimed invention as recited by claim 14.

Contrary to the Examiner's assertions, the configuration program 7 of Adusumilli that creates an initial configuration file 6, propagates user's configuration to all relevant constructs and creates translation tables 3 and C data structures according to input configuration information, is not equivalent to the computer program product, as taught by Applicants and claimed in independent claim 14.

Applicants respectfully submit that the computer program product, as taught by Applicants and claimed in independent claim 14 is not disclosed, nor suggested by Adusumilli. Adusumilli does not disclose, nor suggest the steps of a utilizing sizeof and offsetof functions, defining a length and a location of each parameter of a data structure; and storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures, as taught by Applicants and claimed in independent claim 14.

Lack of novelty can be established only where a prior invention is identical

- 27 -

to (or "anticipates") the invention sought to be patented. "In addition, the prior art reference must be enabling, thus placing the allegedly disclosed matter in the possession of the public." Akzo N.V. v. U.S. Intern. Trade Com'n, 808 F.2d 1471, 1479 (Fed. Cir. 1986).

Thus, independent claim 14 clearly is patentable over Adusumilli. Independent claim 14 is patentable.

Thus, independent claims 1, 10, and 14 are not anticipated by, nor rendered obvious by the Adusumilli patent. Sasagawa et al. fails to disclose and provides no suggestion of storing said length and said location of each said parameter of the data structure within an identifier object in a data structure definition, as taught by Applicants and recited in the independent claims 1, 10, and 14. Sasagawa et al. adds nothing to render obvious the claimed invention, as recited by independent claims 1, 10, and 14, as presented.

D. THE REJECTION OF CLAIMS 2-9 AS BEING UNPATENTABLE OVER ADUSUMILLI AND SASAGAWA et al. SHOULD BE REVERSED

The Board should reverse the rejection of claims 2-9 under 35 USC §103 as being unpatentable over over Adusumilli, U.S. patent 5,870,749 in view of Sasagawa et al., U.S. patent 6,028,863.

35 U.S.C. §103 requires that the invention as claimed be considered "as a whole" when considering whether the invention would have been obvious when it was made. Graham v. John Deere, 383 U.S. 1, 148 USPQ 459, 472 (1966). It is applicants' claimed invention which must be considered as a whole pursuant to 35 U.S.C. §103,

and failure to consider the claimed invention as a whole is an error of law. In order for there to be a prima facie showing of obviousness under 35 U.S.C. §103, it is necessary that the references being combined in an attempt to demonstrate prima facie obviousness must themselves suggest the proposed combination. For a combination of prior art references to render an invention obvious, "[t]here must be some reason, suggestion, or motivation found in the prior art whereby a person of ordinary skill in the field of the invention would make the combination." In re Oetiker, 977 F.2d 1443, 1447, 24 USPQ2D 1443, 1446 (Fed. Cir. 1992). That one must point to some reason, suggestion, or motivation to make a combination is not to say that the teaching must be explicit, but in order to render an invention obvious by the combination of prior art references, the prior art must contain some reason, suggestion, or motivation. It is impermissible to use the inventor's disclosure as a "road map" for selecting and combining prior art disclosures. In Interconnect Planning Corp. v. Feil 774 F.2d 1132, 1143, 227 USPQ 542, 551 (Fed. Cir. 1985), the Federal Circuit noted that "The invention must be viewed not with the blueprint drawn by the inventor, but in the state of the art that existed at the time."

The prior art of record, including the Adusumilli patent and the Sasagawa et al. patent provides no teaching, suggestion or inference in the prior art as a whole or knowledge generally available to one having ordinary skill in the art to achieve the claimed invention. 35 U.S.C. § 103 requires that the invention as claimed be considered "as a whole" when considering whether the invention would have been obvious when it was made. Graham v. John Deere, 383 U.S. 1, 148 USPQ 459, 472

(1966). It is applicant's claimed invention which must be considered <u>as a whole</u>
pursuant to 35 U.S.C. § 103, and failure to consider the claimed invention as a whole is
an error of law.

In the words of the Court of Appeals for the Federal Circuit, "The mere
fact that the prior art may be modified in the manner suggested by the Examiner does
not make the modification obvious unless the prior art suggested the desirability of the
modification." <u>In re John R. Fritch</u>, 972 F.2d 1260, 1266, 23 USPQ2d 1780 (Fed. Cir.
1992). See <u>In re Gordon and Sutherland</u>, 733 F.2d 900, 221 USPQ 1125, 1127 (Fed.
Cir. 1984), <u>Carl Schenck, A.G. v. Nortron Corp.</u>, 713 F.2d 782, 787, 218 USPQ 698,
702 (Fed. Cir. 1983), and <u>In re Sernaker</u>, 702 F.2d 989, 995-96, 217 USPQ 1, 6-7 (Fed.
Cir. 1983).

Representative claim 2 is patentable

Each of the dependent claims 2 and 3 is patentable for the same reasons
discussed above relative to claim 1. Representative claim 2 is submitted to be
separately patentable because claim 2 further defines that the data structure is an ATM

Each of the dependent claims 2 and 3 is patentable for the same reasons
discussed above relative to claim 1. Representative claim 2 is submitted to be
separately patentable because claim 2 further defines that the data structure is an ATM
information element (IE) and wherein the step of utilizing sizeof and offsetof functions,
defining a length and a location of each parameter of a data structure includes the step
of utilizing sizeof and offsetof functions, defining a length and a location of each data
parameter of said ATM information element (IE).

Applicant respectfully submits that the prior art descriptions of Adusumilli and the Sasagawa et al. fall short of applicant's invention, and the subject matter of the claimed invention as recited in claim 2 would not have been obvious to one of ordinary skill in the art in view of the references of record. Further in the cited references, there is no hint of defining a length and a location of each data parameter of said ATM information element (IE), as taught and claimed by Applicants. A combination of all the teachings of the references of record would not achieve the claimed invention as recited by claim 2.

In order for there to be a prima facie showing of obviousness under 35 U.S.C. §103, it is necessary that the references being combined in an attempt to demonstrate prima facie obviousness must themselves suggest the proposed combination. It is insufficient to establish obviousness that the separate elements of the invention existed in the prior art, absent some teaching or suggestion, in the prior art, to combine the elements. Arkie Lures, Inc. v. Gene Larew Tackle, Inc., 119 F.3d 953, 957, 43 USPQ2d 1294, 1297 (Fed. Cir. 1997).

The motivation, suggestion or teaching may come explicitly from statements in the prior art, the knowledge of one of ordinary skill in the art, or, in some cases the nature of the problem to be solved. In re Kotzab, 217 F.3d 1365, 1370, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000). Hindsight in impermissible when an examiner rejects an application in reliance upon teachings not drawn from any prior art disclosure, but from the applicant's own disclosure. In re Dembiczak, 175 F.3d 994, 998, 50 USPQ2d 1614, 1616 (Fed. Cir. 1999). Broad conclusory statements standing

alone are not "evidence." Id.

The test for obviousness is what the combined teachings of the references would have suggested to one of ordinary skill in the art. See In re Young, 927 F.2d 588, 591, 18 USPQ2d 1089, 1091 (Fed. Cir. 1991). The mere fact that the prior art could be modified so as to result in the combination defined by the claims would not have made the modification obvious unless the prior art suggests the desirability of the modification. See In re Gordon and Sutherland, 733 F.2d 900, 221 USPQ 1125, 1127 (Fed. Cir. 1984), Carl Schenck, A.G. v. Nortron Corp., 713 F.2d 782, 787, 218 USPQ 698, 702 (Fed. Cir. 1983), and In re Sernaker, 702 F.2d 989, 995-96, 217 USPQ 1, 6-7 (Fed. Cir. 1983).

In a proper obviousness determination, "whether the changes from the prior art are 'minor', . . . the changes must be evaluated in terms of the whole invention, including whether the prior art provides any teaching or suggestion to one of ordinary skill in the art to make the changes that would produce the patentee's . . . device." This includes what could be characterized as simple changes, as in In re Gordon and Sutherland, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1983) (Although a prior art device could have been turned upside down, that did not make the modification obvious unless the prior art fairly suggested the desirability of turning the device upside down.).

In Re Fritch 972 F.2d at 1266, 23 USPQ2d at 1780 (Fed. Cir. 1992), states: "[I]t is impermissible to use the claimed invention as an instruction manual or 'template' to piece together the teachings of the prior art so that the claimed invention is rendered obvious. ... This court has previously stated that '[o]ne cannot use hindsight

reconstruction to pick and choose among isolated disclosures in the prior art to deprecate the claimed invention."

Applicants acknowledges that the ATM information element is known, and that Sasagawa. Applicants submit that the subject matter of the claimed invention as recited in claim 2 would not have been obvious to one of ordinary skill in the art in view of the references of record. No hint is found in the references of record and the references of record do not suggest the step of defining a length and a location of each data parameter of said ATM information element (IE), as taught and claimed by Applicants in claim 2.

Each of the dependent claims 2 and 3 is patentable.

Representative claim 5 is patentable

Each of claims 5, 6, 8 and 9 is submitted to be patentable for the same reasons set forth above in connection with claim 1. Representative claim 5 is separately patentable further defining the invention of claims 1 and 3, reciting that the ATM information element (IE) is a Connection Identifier IE and further includes the step of utilizing sizeof and offsetof functions, defining a length and a location of a virtual path connection identifier (VPCI) parameter.

Rejections based on § 103 must rest on a factual basis with these facts being interpreted without hindsight reconstruction of the invention from the prior art. The Examiner may not, because of doubt that the invention is patentable, resort to speculation, unfounded assumption or hindsight reconstruction to supply deficiencies in the factual basis for the rejection. See In re Warner, 379 F.2d 1011, 1017, 154 USPQ

173, 178 (CCPA 1967), cert. denied, 389 U.S. 1057 (1968).

Utilizing sizeof and offsetof functions, defining a length and a location of a virtual path connection identifier (VPCI) parameter, as taught and claimed by Applicants in claim 5, is neither disclosed nor suggested by the references of record, including Adusumilli and the Sasagawa et al. The fact that the disclosed process of Adusumilli is described to support representation of Managed Object attribute data in a space-efficient manner or in a form suitable for efficient database access by CMIP a well as non-CMIP applications, adds nothing to suggest the invention as defined in claim 5.

Thus, each of claims 5, 6, 8 and 9 is further patentable over the references of record.

Representative claim 7 is patentable

Each of claims 4, and 7 is submitted to be patentable for the same reasons set forth above in connection with claim 1. Further representative claim 7 is separately patentable further defining the invention of claim 1 reciting that the step of storing said length and said location of each said parameter of the data structure within an identifier object in a data structure definition includes the steps of storing said length and said location of said preferred/exclusive parameter in a preferred/exclusive parameter identifier object in said data structure definition. Only applicants teach storing said length and said location of each said parameter of the data structure within an identifier object in a data structure definition. Only applicants teach storing said length and said location of said preferred/exclusive parameter in a preferred/exclusive parameter identifier object in said data structure definition.

Applicants teach the subject matter of representative claim 7, where storing said length and said location of said preferred/exclusive parameter in a preferred/exclusive parameter identifier object in said data structure definition, for example, as illustrated in FIGS. 5 and 6.

The claimed feature is not shown nor suggested in total combination of the references of record, including Adusumilli and Sasagawa et al. Thus, each of claims 4, and 7 is further patentable over the references of record.

Hindsight in impermissible when an examiner rejects an application in reliance upon teachings not drawn from any prior art disclosure, but from the applicant's own disclosure. In re Demiski, 796 F.2d 236,443, 230 USPQ2d 313, 316 (Fed. Cir. 1986); W.L. Gore & Assocs. Inc. V. Garlock, Inc., 721 F.2d 1540, 1553, 220 USP1 303, 313 (Fed. Cir. 1984), cert. Denied, 469 U.S. 851 (1984).

That one must point to some reason, suggestion, or motivation to make a combination is not to say that the teaching must be explicit, but in order to render an invention obvious by the combination of prior art references, the prior art must contain some reason, suggestion, or motivation. It is impermissible to use the inventor's disclosure as a "road map" for selecting and combining prior art disclosures. In Interconnect Planning Corp. v. Feil 774 F.2d 1132, 1143, 227 USPQ 542, 551 (Fed. Cir. 1985), the Federal Circuit noted that "The invention must be viewed not with the blueprint drawn by the inventor, but in the state of the art that existed at the time."

E. CONCLUSION

Claims 1-14 are patentable over all the references of record and are

neither anticipated not rendered obvious by the Adusumilli and Sasagawa et al.

patents. Each of the claims 1-14 is patentable and the Examiner's rejections should be

reversed.

It is respectfully requested that the final rejection be reversed.

Respectfully submitted,
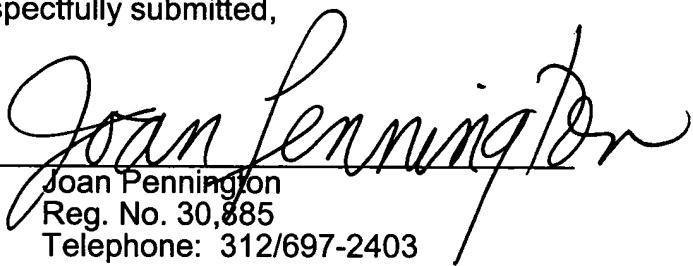
By _____
   Joan Pennington
   Reg. No. 30,885
   Telephone: 312/697-2403

(8) CLAIMS APPENDIX

CLAIMS ON APPEAL

1. (previously presented)   A computer-implemented method for parsing and generating data structures for use by data processing applications in a computer system comprising the steps of:

utilizing sizeof and offsetof functions, defining a length and a location of each parameter of a data structure; and

storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures.

2. (original)   A method for parsing and generating data structures as recited in claim 1 wherein the data structure is an ATM information element (IE) and wherein the step of utilizing sizeof and offsetof functions, defining a length and a location of each parameter of a data structure includes the step of utilizing sizeof and offsetof functions, defining a length and a location of each data parameter of said ATM information element (IE).

3. (previously presented)   A method for parsing and generating data structures as recited in claim 2 wherein said ATM information element (IE) is a Connection Identifier IE and wherein the step of utilizing sizeof and offsetof functions, defining a length and a location of each parameter of a data structure includes the step of utilizing sizeof and offsetof functions, defining a length and a location of each data parameter of said Connection Identifier IE.

4. (original)   A method for parsing and generating data structures as recited in claim 3 includes the step of utilizing sizeof and offsetof functions, defining a length and a location of a preferred/exclusive parameter.

5. (original)   A method for parsing and generating data structures as recited in claim 3 includes the step of utilizing sizeof and offsetof functions, defining a length and a location of a virtual path connection identifier (VPCI) parameter.

6. (original)   A method for parsing and generating data structures as recited in claim 3 includes the step of utilizing sizeof and offsetof functions, defining a length and a location of a virtual channel identifier (VCI) parameter.

7. (original)   A method for parsing and generating data structures as recited in claim 4 wherein the step of storing said length and said location of each said parameter of the data structure within an identifier object in a data structure definition includes the steps of storing said length and said location of said preferred/exclusive parameter in a preferred/exclusive parameter identifier object in said data structure definition.

8. (original)   A method for parsing and generating data structures as recited in claim 5 wherein the step of storing said length and said location of each said parameter of the data structure within an identifier object in a data structure definition includes the steps of storing said length and said location of said virtual path connection identifier (VPCI) parameter in a VPCI parameter identifier object in said data structure definition.

9. (original)   A method for parsing and generating data structures as recited in claim 6 wherein the step of storing said length and said location of each said parameter of the data structure within an identifier object in a data structure definition includes the

steps of storing said length and said location of said virtual channel identifier (VCI)

parameter in a VCI parameter identifier object in said data structure definition.

10. (previously presented)   A compiler and platform independent

framework for parsing and generating data structures used by data processing

applications in a computer system comprising:

means for defining a length and a location of each parameter of a data structure

utilizing sizeof and offsetof functions; and

means for storing said defined length and said defined location of each said

parameter of the data structure within an identifier object in a data structure definition

used for parsing and generating data structures.

11. (original) A compiler and platform independent framework for parsing and

generating data structures as recited in claim 10 is used by procedural table-driven or

object rules-driven methods for parsing and generating data structures.

12. (original)   A compiler and platform independent framework for parsing

and generating data structures as recited in claim 10 is used for parsing and generating

of protocol data units (PDUs) in data communication messages.

13. (original)   A compiler and platform independent framework for parsing

and generating data structures as recited in claim 10 is used for parsing and generating

of control code for writing and reading headers for data storage.

14. (previously presented)   A computer program product for parsing and

generating data structures for use by data processing applications in a computer

system, said computer system having a processor; a memory controller coupled to said

processor by a system bus; a main memory coupled to said memory controller; said computer program product including a plurality of computer executable instructions stored on a computer readable medium, wherein said instructions, when executed by said computer system, cause said computer system to perform the steps of:

utilizing sizeof and offsetof functions, defining a length and a location of each parameter of a data structure; and

storing said defined length and said defined location of each said parameter of the data structure within an identifier object in a data structure definition used for parsing and generating data structures.

Serial No. 09/829,834

(9) EVIDENCE APPENDIX
DRAWINGS OF INVENTION
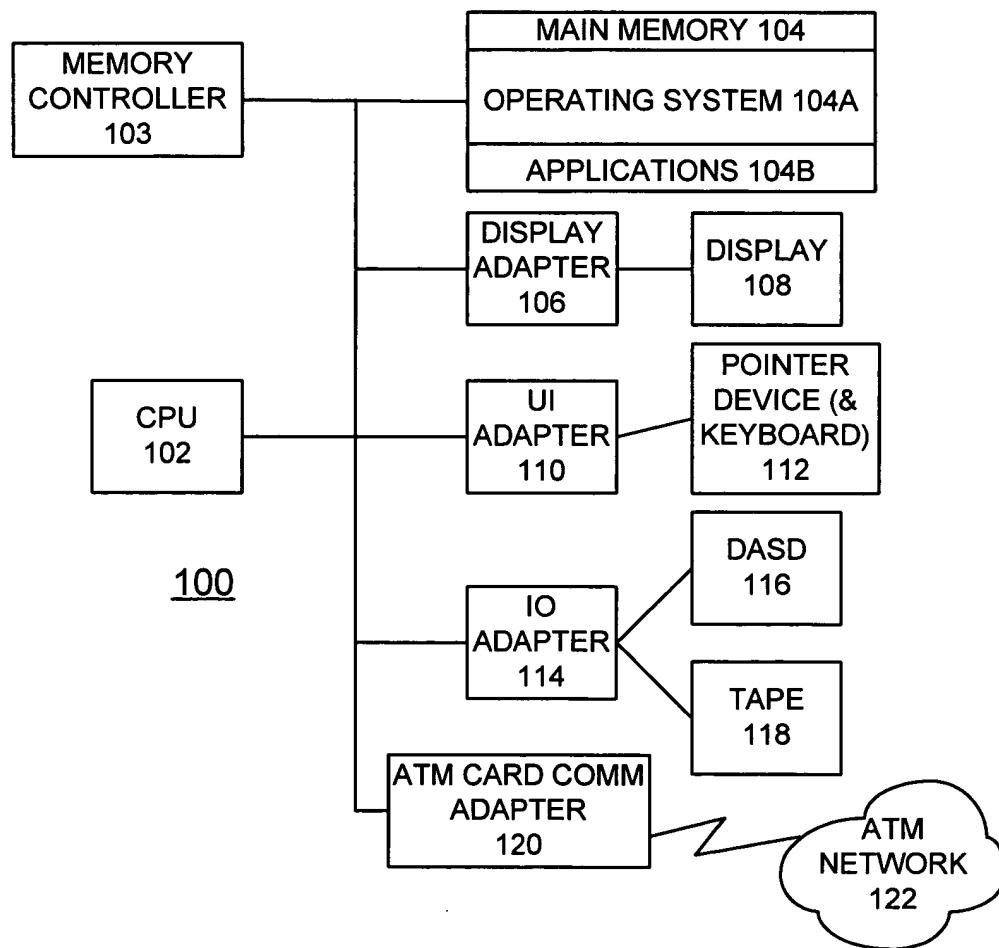
FIG. 1
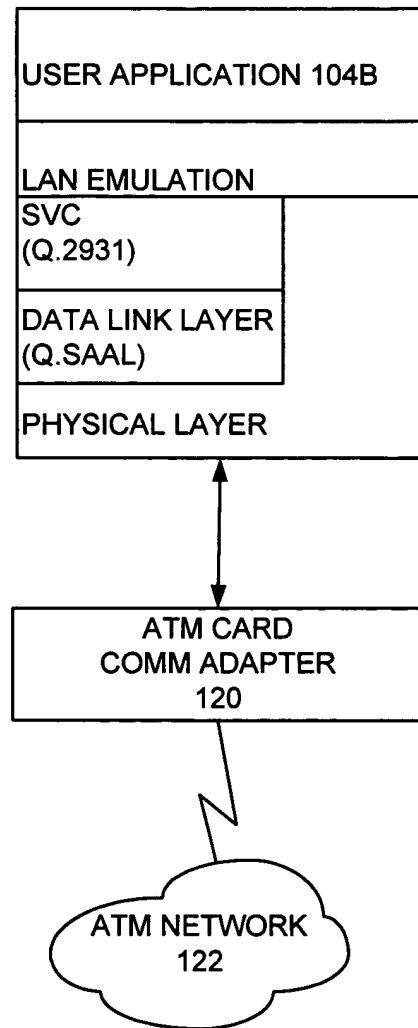
FIG. 2A

CONNECTION IDENTIFIER INFORMATION ELEMENT (IE) 200

Connection Identifier							
0	1	0	1	1	0	1	0
information element identifier = 0X5A							

1 ext	Coding standard	IE Instruction Field	
Length of Connection Identifier contents			
Length of Connection Identifier contents (cont.)			
1 ext	spare 0 0	VPassociated signaling 0 1	preferred exclusive
VPCI			
VPCI (cont.)			
VCI			
VCI (cont.)			

FIG. 2B

AtmSvcConnIdle Rules 202

Information element identifier = 0x5A;
pointer to 1st data rule =

grp 5: an Extension bit Group 204

initialize byte to 0x08;
pointer to 1st data rule =

prefExclConnIdleParm_; sub byte data

mask defining location within byte =0x07;
allowed values = {0x00, 0x01, 0x04;
length and offset of prefd_exclus within conn_id_struct

pointer to next data rule = NULL;

pointer to next group rule =

grp 6: a fixed length group 206

pointer to 1st data rule =
vpciConnIdleParm_ : two byte data

allowed range of values = 0 →255
length and offset of vpci within conn_id_struct
pointer to next data rule = NULL;

pointer to next group rule =

grp 7: a fixed length group 208

pointer to 1st data rule =
vciConnIdleParm : two byte data

allowed range of values = 32 →65535
length and offset of vci within conn_id_struct
pointer to next data rule = NULL;

pointer to next group rule = NULL;

FIG. 2C
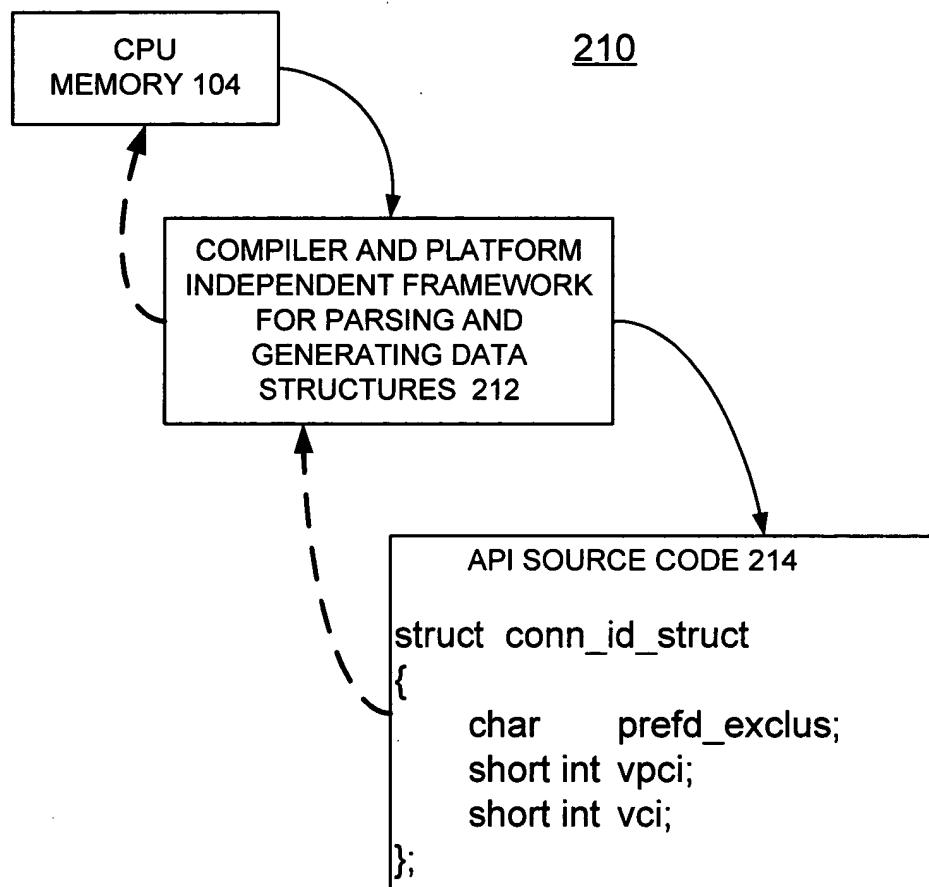
FIG. 2D

300

	0	1 2	3
0	prefd_exclus	vpci	vci
1	vci		

FIG. 3

400

structure field 402	sizeof() 404	offsetof() 406
prefd_exclus	1	0
vpci	2	1
vci	2	3

FIG. 4

500

	0	1	2	3
0	prefd_exclus	x	vpci	
1	vci		x	x

FIG. 5

600

structure field 602	sizeof() 604	offsetof() 606
prefd_exclus	1	0
vpci	2	2
vci	2	4

FIG. 6

700

	0	1	2	3
0	prefd_exclus	x	x	x
1	vpci		x	x
2	vci		x	x

FIG. 7

800

structure field 802	sizeof() 804	offsetof() 806
prefd_exclus	1	0
vpci	2	4
vci	2	8

FIG. 8

FIG. 9

SOURCE CODE 900

```
struct conn_id_struct
{
    void*         next;
    char          prefd_exclus;
    short int     vpci;
    short int     vci;
};
```

FIG. 15

1000

	0	1	2	3
0	next			
1	prefd_exclus	x	vpci	
2	vci		x	x
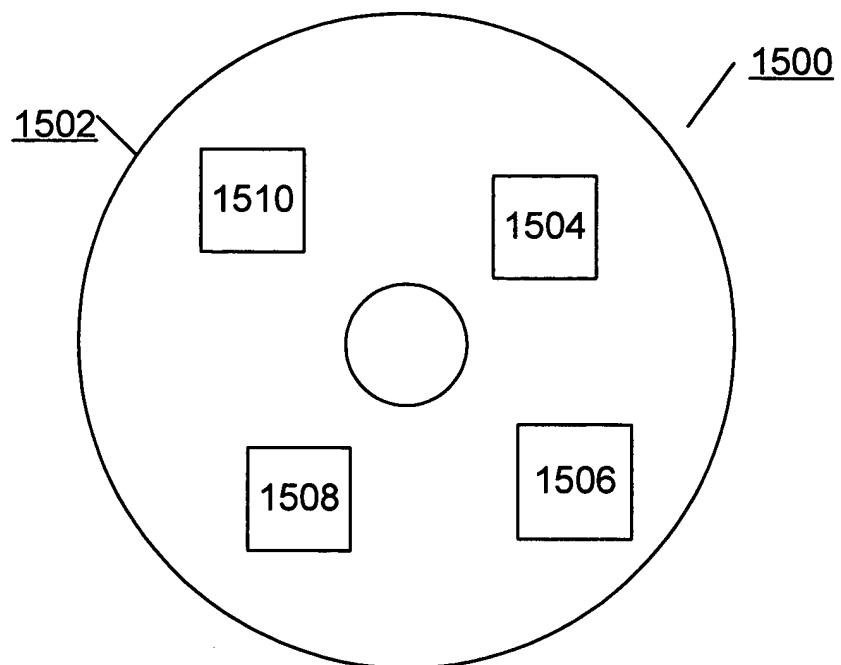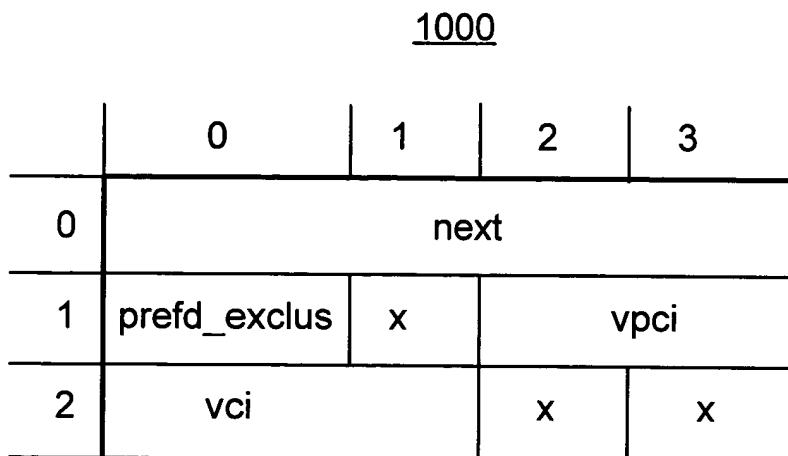
FIG. 10

1100

structure field 1102	sizeof() 1104	offsetof() 1106
next	4	0
prefd_exclus	1	4
vpci	2	6
vci	2	8

FIG. 11

1200

	0	1	2	3
0			next	
2				
3	prefd_exclus	x	vpci	
4	vci		x	x
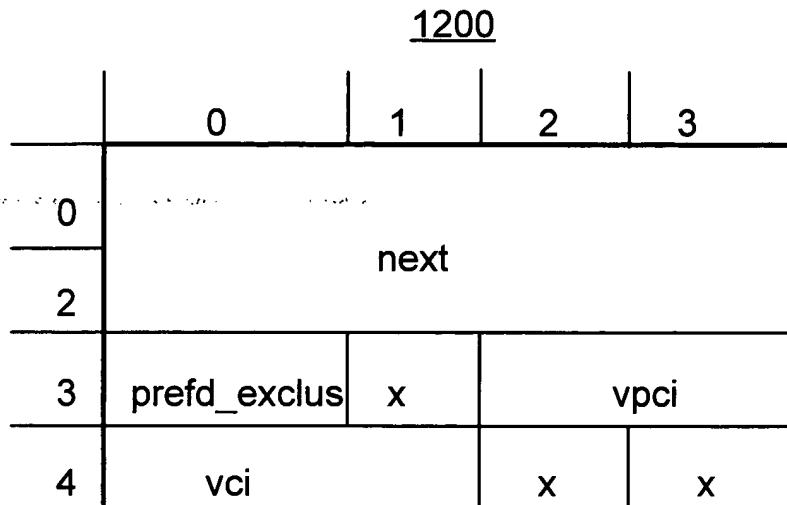
FIG. 12

1300

structure field 1302	sizeof() 1304	offsetof() 1306
next	8	0
prefd_exclus	1	8
vpci	2	10
vci	2	12

FIG. 13

FIG. 14A

```
┌─────────────────────────┐
│  BUILD HEADER USING THE IE │
│     IDENTIFIER 1400        │
└─────────────────────────┘
            │
            ▼
┌─────────────────────────┐
│  GET 1ST GROUP RULE 1402 │
└─────────────────────────┘
      C ──────────►│
            ▼
┌─────────────────────────┐
│ GET 1ST DATA RULE IN GROUP 1404 │
└─────────────────────────┘
      B ──────────►│
            ▼
```

FIXED LENGTH GROUP ◄─────── TYPE OF GROUP? 1406 ───────► EXTENSION BIT GROUP

```
                              ┌─────────────────────────┐
                              │   SET PDU BYTE TO        │
                              │  INITIAL VALUE 1408      │
                              └─────────────────────────┘
            ▼
┌─────────────────────────┐
│   GET 1ST DATA RULE IN   │
│      GROUP 1410          │
└─────────────────────────┘
            │
            ▼
```

TWO-BYTE DATA ◄─────── TYPE OF DATA RULE? 1412 ───────► SUB-BYTE DATA

| READ DATA FROM API STRUCTURE USING LENGTH AND OFFSET STORED IN THIS DATA RULE. WRITE THIS DATA INTO NEXT 2 BYTES OF PDU. ADVANCE PDU BUFFER POINTERS 1414 | READ DATA FROM API STRUCTURE USING LENGTH AND OFFSET STORED IN THIS DATA RULE. WRITE THIS DATA IN BITS DEFINED BY MASK STORED IN THIS DATA RULE. ADVANCE PDU BUFFER POINTERS IF THIS WAS LAST DATA RULE FOR THIS GROUP 1416 |

- 52A

FIG. 14B

(10) RELATED PROCEEDINGS APPENDIX

None.